APLIKASI KAMUS ISTILAH AKUNTANSI MENGGUNAKAN METODE *KNUTH MORRIS PRATT*

Eva Darnila
Teknik Informatika, Universitas Malikussaleh Lhokseumawe, Aceh Utara
eva_daud@yahoo.com

ABSTRACT

The dictionary accounting terms is very benefit the accountants and the parties are engaged in economic, because it can it easier for accountants in searching vocabulary eliminating the need using the general dictionaries. Therefore, it takes a dictionaries the term accounting application integrated with mobile tools will greatly simplify the user in searching meaning of the word of a term in the field of accounting. The process of finding the term application using the Knuth morris pratt the also a method of string matching. Knuth Morris pratt method using the shift characters based on because Knuth Morris pratt method information obtained before the process (pre-process). Knuth method morris pratt do a considerable shift characters using the function limitation rules (border function).

Keywords : String Matching, Knuth Morris Pratt

INTISARI

Kamus istilah akuntansi sangat bermamfaat bagi para akuntan dan pihak yang bergerak dalam bidang ekonomi, karena dapat mempermudah para akuntan dalam mencari kosa kata sehingga tidak perlu menggunakan kamus umum. Oleh karena itu dibutuhkan aplikasi kamus istilah akuntansi yang terintegrasi dengan perangkat selular akan sangat mempermudah pengguna dalam mencari arti kata dari suatu istilah dalam bidang akuntansi. Proses pencarian aplikasi istilah menggunakan metode *knuth morris pratt* yang juga merupakan metode pencocokan *string*. Metode *knuth morris pratt* menggunakan pergeseran karakter berdasarkan informasi yang diperoleh sebelum proses dilakukan (*pre-process*). Metode *knuth morris pratt* dinilai cukup efektif dalam melakukan pencarian, dikarenakan metode *knuth morris pratt* melakukan pergeseran karakter yang cukup jauh dengan menggunakan aturan fungsi batasan (*border function*).

Kata kunci : *String Matching, Knuth Morris Pratt*

I. PENDAHULUAN

Akuntansi merupakan satu bidang ilmu yang dalam pembelajarannya memiliki banyak istilah-istilah yang sulit dipahami. Sebagai alat bantu untuk memahami istilah-istilah tersebut, maka kamus istilah akuntansi dirasa perlu sebagai penunjang pembelajaran. Kamus pada umumya memiliki bentuk berupa buku tebal dengan proses pencarian kata yang harus dilakukan secara manual dengan cara menyisir kata satu per satu.

Kamus istilah dibuat khusus untuk menjelaskan berbagai istilah dari bidang-bidang tertentu. Saat ini terdapat banyak kamus istilah yang tersedia berbentuk buku. Penggunaan kamus dalam bentuk buku dirasa kurang efektif, mengingat pencarian masih dilakukan secara manual dan membutuhkan waktu yang lama. Kamus berbentuk buku juga sulit dibawa karena berat dan tebal. Selain itu kamus istilah juga sedikit sulit untuk didapatkan terutama bagi mereka yang tinggal di kota kecil dan tidak memiliki sumber daya toko buku yang lengkap.

Penggunaan kamus model lama tentu tidak sejalan dengan gaya hidup modern saat ini, dimana segala sesuatunya dituntut untuk berjalan dengan cepat dan praktis. Oleh karena itu diperlukan aplikasi kamus istilah yang dapat terintegrasi dengan perangkat selular yang dapat memudahkan pengguna melakukan proses pencarian secara instan tanpa harus menyisir kata satu per satu. Sistem operasi Android dipilih sebagai dasar dari aplikasi ini dikarenakan Android merupakan sistem operasi selular yang cukup banyak digunakan saat ini.

Aplikasi kamus dalam berntuk *smartphone* merupakan solusi yang dapat menjawab kekurangan dari penggunaan kamus berbentuk buku, kamus juga dapat digunakan dimanapun

dan kapanpun serta dapat memudahkan penggunanya. Untuk membuat aplikasi kamus dalam bentuk *smartphone* tentu dibutuhkan sebuah Metode untuk melakukan proses pencarian data. Proses pencarian aplikasi kamus istilah membutuhkan suatu metode pencarian. Metode pencarian dapat dilakukan dengan menggunakan proses *string matching* atau pencocokkan *string*. S*tring matching* merupakan proses pencocokan *string* dari *string* yang lebih pendek (*pattern*) terhadap *string* yang lebih panjang (*text*). Proses pencocokan *string* dapat dilakukan dengan beberapa metode, salah satunya adalah metode *knuth morris pratt*. Metode *knuth morris pratt* dinilai cukup efektif dalam melakukan pencarian dikarenakan metode *knuth morris pratt* melakukan pergeseran karakter yang cukup jauh dengan menggunakan aturan fungsi batasan (*border function*). Metode *knuth morris pratt* menggunakan pergeseran karakter berdasarkan informasi yang diperoleh sebelum proses dilakukan (*pre-process*). Diharapkan dengan adanya aplikasi kamus istilah Akuntansi berbasis Android ini dapat menjawab kekurangan dari kamus berbentuk buku serta dapat menjadi salah satu referensi belajar dalam membantu pengguna untuk menemukan arti dari istilah-istilah yang dibutuhkan.

II. METODE PENELITIAN

A. Kamus

Kamus adalah buku acuan yang memuat kata dan ungkapan yang disusun menurut abjad beserta keterangan tentang makna, pemakaian atau terjemahannya. Kamus disusun sesuai dengan abjad dari A-Z dengan tujuan untuk memudahkan pengguna kamus dalam mencari istilah yang diinginkannya dengan cepat. Kamus memiliki kegunaan untuk memudahkan penggunanya dalam mencari istilah-istilah yang belum dipahami maknanya [1].

Kamus istilah termasuk kategori kamus khusus karena merujuk kepada kamus yang mempunyai fungsi khusus. Kamus ini berisi istilah-istilah khusus dalam bidang tertentu, fungsinya untuk kegunaan ilmiah (Kurniasih dan Ema : 2014), sedangkan kamus elektronik memanfaatkan media tertentu baik aplikasi desktop, web, maupun kamus berbasis *mobile*. Fungsi dari kamus elektronik sama dengan kamus berbentuk buku, hanya saja dengan menggunakan kamus elektronik pencarian dapat dilakukan dengan lebih cepat.

B. Pencarian Data (*Searching*)

Algoritma pencarian (*searching algorithm*) adalah algoritma yang menerima sebuah argumen kunci dan dengan langkah-langkah tertentu akan mencari rekaman dengan kunci tersebut. Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan (*successful*) atau tidak ditemukan (*unsuccessful*). Pencarian data (*searching*) merupakan proses yang fundamental dalam pengolahan data. Proses pencarian adalah menemukan nilai tertentu di dalam sekumpulan data bertipe sama (tipe dasar atau tipe bentukan). Aktivitas yang berkaitan dengan pengolahan data sering didahului dengan proses pencarian [2].

C. Android

Android adalah sebuah sistem operasi *mobile* yang berbasiskan pada versi modifikasi dari linux. Sistem operasi ini pertama kali dikembangkan oleh perusahaan Android Inc. Nama perusahaan inilah yang pada akhirnya digunakan sebagai nama proyek sistem operasi mobile tersebut, yaitu sistem operasi Android [3].

Android Inc didirikan oleh "Andy Rubin, Rich Milner, Nick Sears dan Chris White" pada tahun 2003. Pada Agustus Tahun 2005, sebagai bagian dari strategi untuk memasuki pasar *mobile* Google Inc, membeli Android dan mengambil alih proses pengembangannya sekaligus Tim Developer Android. Google menginginkan Android untuk menjadi sistem operasi *Open Source Apache* yang berarti setiap orang bebas untuk menggunakan dan mengunduh *source code* Android secara penuh. Kemudian untuk mengembangkan Android dibentuklah *Open Handset Alliance,* konsorsium dari 34 perusahaan peranti keras, peranti lunak dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile dan Nividia. Pada saat perilisan perdana

Android 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat mobile. Di lain pihak Google merilis kode-kode Android di bawah lisensi *Apache*, sebuah lisensi perangkat lunak dan *open platfrom* perangkat seluler [4].

D. Bahasa Pemrograman JAVA

Bahasa pemograman merupakan suatu instruksi pada komputer yang terdiri dari sintaks untuk mendefinisikan program komputer, bahasa pemograman membatu seorang programmer dalam menentukan data mana yang akan diolah komputer, penyimpanan data dan langkah apa yang akan diambil dalam berbagai kondisi [5].

Sedangkan *Java* merupakan sebuah *platform* sekaligus bahasa pemograman tingkat tinggi yang mempunyai kriteria sederhana, berorientasi objek, terdistribusi, dinamis, aman dan lainnya. Dikembangkan dengan model yang mirip dengan C++ namun lebih mudah dipakai juga memiliki *platform* Independent yang dapat dijalankan pada sistem operasi apapun. Pada pemograman *Java*, seluruh *source code* pertama kali ditulis dalam file teks biasa yang akan berubah menjadi ekstensi (.Java). Lalu seluruh file *source code* tersebut akan di compile menjadi *bytecode* dengan ekstensi (.class) oleh Javac Compiler. *Bytecode* tersebut dapat dieksekusi di tiap *platform* menggunakan *Java Virtual Machine* (JVM) atau *Java Runtime* sebagai *interprenter*. *Java Virtual Machine* dapat diartikan sebagai salah satu komponen dari *Java platform* selain *Aplication Programming Interface* (API) , sedangkan *platform* sendiri merupakan *hardware* juga *software* tempat sebuah program dapat dijalankan seperti Windows, Linux dan lainnya. *Compiler* untuk program java adalah JDK yang diproduksi oleh Sun Microsystem. JDK menyediakan dua program utama yaitu *Javac* untuk mengcompile kode sumber dan *Java* sebagai program untuk meluncurkan aplikasi.

E. UML (Unified Modelling Language)

UML adalah bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metedologi tertentu, meskipun pada kenyataannya UML paling banyak digunkan pada metodologi berorientasi objek [6].

F. Pencocokan String (*String Matching*)

String matching atau yang sering disebut juga dengan pencocokan *string* adalah algoritma untuk melakukan pencarian semua kemunculan *string* pendek yang disebut *pattern* di *string* yang lebih panjang yang disebut *text*. Pencocokkan *string* merupakan permasalahan paling sederhana dari semua permasalahan *string* lainnya, dan dianggap sebagai bagian dari pemrosesan data, pengkompresian data, analisis leksikal, dan temu balik informasi. Teknik untuk menyelesaikan permasalahan pencocokkan *string* biasanya akan menghasilkan implikasi langsung ke aplikasi *string* lainnya [7].

Pencarian *string* juga biasa digunakan untuk mencari pola bit dalam sejumlah besar *file binary*. Contoh permasalahannya adalah dalam program *text editor* seperti Microsoft Word, atau dalam lingkup yang lebih besar adalah pencarian web seperti Google, Bing, dan Yahoo. Proses pencocokan *string* yang merupakan bagian dalam proses pencarian *string* memegang peranan penting untuk mendapatkan dokumen yang sesuai dengan kebutuhan informasi. Pencocokan *string* secara garis besar dapat dibedakan menjadi dua yaitu pencocokan *string* secara eksak/sama persis (*exact string matching*) dan pencocokan *string* berdasarkan kemiripan (*inexact string matching/fuzzy string matching*). Pencocokan *string* berdasarkan kemiripan masih dapat dibedakan menjadi dua yaitu berdasarkan kemiripan penulisan (*approximate string matching*) dan berdasarkan kemiripan ucapan (*phonetic string matching*). Algoritma pencarian *string* atau sering disebut juga pencocokan *string* adalah algoritma untuk melakukan

pencarian semua kemunculan *string* pendek *pattern*[0..n − 1] yang disebut *pattern* di *string* yang lebih panjang teks[0..m − 1] yang disebut teks [8].

G. Metode Knuth Morris Pratt

Algoritma pencocokan *string* yang mempunyai kinerja bagus adalah *knuth morris pratt* dan algoritma *boyer moore*. kedua algoritma ini popular digunakan pada editor teks, mesin pencari, analisis citra, dan sebagainya. *Search engine* atau mesin pencari adalah program komputer yang dirancang untuk membantu seseorang menemukan *file* yang disimpan dalam komputer, baik itu dalam sebuah server umum di web atau dalam komputer itu sendiri. Mesin pencari memungkinkan kita untuk meminta *content media* dengan kriteria yang spesifik (biasanya yang berisi kata atau frasa yang telah ditentukan) dan memperoleh daftar *file* yang memenuhi kriteria tersebut [9].

Algoritma *knuth morris pratt* adalah algoritma pencarian *string* yang mencari dengan cara menghitung ketika ketidakcocokan ditemukan, dari ketidakcocokan tersebut akan dihitung dari mana pencarian selanjutnya sebaiknya dimulai. Algoritma *knuth morris pratt* juga menggunakan fungsi pembatas (*border function)* yang digunakan untuk menghitung urutan ke berapa perbandingan harus dilakukan. *Border function* dihitung dengan menghitung panjang *prefix* yang ada disebuah *pattern* yang sama dengan *suffix*-nya.

Algoritma *knuth morris pratt* dikembangkan oleh D. E. Knuth, bersama dengan J. H. Morris dan V. R. Pratt. Untuk pencarian *string* dengan menggunakan algoritma *brute force*, setiap kali ditemukan ketidakcocokan *pattern* dengan *text*, maka *pattern* akan digeser satu karakter ke kanan. Sedangkan dalam algoritma *knuth morris pratt*, algoritma *knuth morris pratt* akan memelihara informasi yang digunakan untuk menentukan jumlah pergeseran. Algoritma *knuth morris pratt* nantinya akan menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh, dan tidak hanya satu karakter, sehingga proses pencarian menggunakan algoritma *knuth morris pratt* menjadi lebih cepat dibanding algoritma *brute force*.

Secara sistematis, langkah-langkah yang dilakukan algoritma *knuth morris pratt* pada saat mencocokkan *string*, yaitu :

1. Algoritma *knuth morris pratt* memulai mencocokkan *pattern* pada awal *text*. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pada *pattern* dengan karakter di *text* yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
   a. Karakter di *pattern* dan di *text* yang dibandingkan tidak cocok (*mismatch*).
   b. Semua karakter di *pattern* cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
2. Algoritma kemudian menggeser *pattern* dan mengulangi pergeseran tersebut sampai *pattern* berada di ujung *text*. Algoritma ini menemukan semua kemunculan dari *pattern* dengan panjang *n* di dalam *text* dengan panjang *m* dengan kompleksitas waktu O(*m*+*n*). Algoritma ini hanya membutuhkan O(*n*) ruang dari *memory internal*. semua besaran O tersebut tidak tergantung pada besarnya ruang alphabet.

III. HASIL DAN PEMBAHASAN

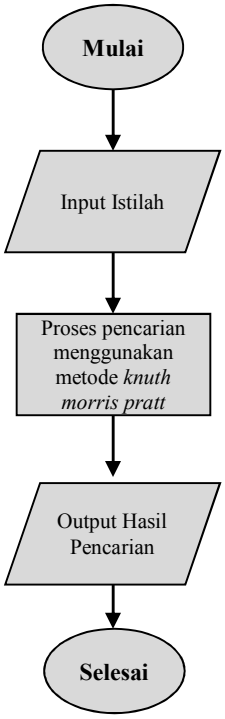
A. Analisa Sistem

Pencarian kata istilah akuntansi saat ini dilakukan dengan menggunakan buku kamus istilah akuntansi. Penggunaan buku kamus akuntansi dalam pencarian kata-kata istilah akuntansi masih menyulitkan pengguna karena pencarian kata-kata istilah dilakukan secara manual yaitu dengan cara melihat daftar istilah pada kamus akuntansi satu persatu-satu. Dalam proses pencarian istilah menggunakan kamus berbentuk buku seorang pengguna harus melakukan pencarian istilah secara manual, pencarian istilah secara manual cenderung membutuhkan waktu yang lama, selain itu pengguna juga dituntut harus melakukan pencarian dengan teliti, mengingat kata-kata yang dimuat dalam kamus disusun berurut dan memiliki kemiripan kosa kata. Pencarian istilah menggunakan kamus juga cenderung melelahkan dan membosankan, mengingat saat ini orang-orang lebih senang untuk

menggunakan teknologi dalam melakukan aktivitasnya. Namun seiring dengan perkembangan ilmu pengetahuan dan teknologi maka kamus dalam bentuk buku dapat dirubah menjadi sebuah aplikasi kamus yang dapat digunakan pada *smartphone* Android. Oleh karena itu penelitian ini bertujuan untuk merancang dan membangun sebuah aplikasi kamus istilah Akuntansi berbasis Android dengan mengimplementasikan *Knuth Morris Pratt* pada proses pencarian istilah.

B. Skema Sistem Proses Pencarian

Skema proses pencarian menggunakan algoritma knuth morris pratt adalah sebagai berikut:



Gambar 1. Flowchart proses pencarian

Proses pencarian dimulai dengan memasukkan istilah yang ingin dicari pada *text field* yang tersedia, dan kemudian saat tombol Cari ditekan, proses pencarian menggunakan metode *knuth morris pratt* pun berlangsung. Proses pencarian dilakukan dengan membaca istilah yang dimasukkan sebagai *pattern* dan mencocokkannya dengan *text* berupa kata-kata istilah yang ada pada *database*. Hasil pencarian berupa dua kemungkinan, kemungkinan pertama yaitu jika istilah yang dicari dapat ditemui, dan aplikasi akan menampilkan arti kata dari istilah yang dimasukkan. Dan kemungkinan kedua yaitu saat istilah tidak dapat dijumpai, maka aplikasi akan memberikan pemberitahuan bahwa kata yang dicari tidak ditemui.

C. Analisa Metode Knuth Morris Pratt

Metode *knuth morris pratt* bekerja dengan langkah-langkah sebagai berikut:

1. Diketahui *pattern* 'ATIKA' dengan panjang i, dan *text* 'SALAM INFORMATIKA' dengan panjang j.
2. Langkah pertama yaitu membuat tabel *next* yang akan menentukan sejauh mana pergeseran akan dilakukan ketika ketidakcocokan (*mismatch*) ditemui. Tabel *next* ditentukan dengan cara menghitung panjang *prefix pattern* yang sesuai dengan *suffix* atau kemunculan karakter yang sama pada *pattern*.
3. Nilai *suffix* dihitung dengan cara memberi nilai yang sesuai dengan kemunculan karakter yang sama pada *pattern*. Jika karakter belum pernah muncul pada *pattern*, nilainya adalah 0. Jika karakter sudah pernah muncul dalam *pattern*, maka bernilai 1. Jika karakter memiliki kombinasi P[i-n,...,i-1,i] yang sama dengan kombinasi karakter sebelumnnya pada *pattern*, maka bernilai 2, 3, dan seterusnya.

Sebagai contoh, untuk *pattern* X[n]=PANPANCA dengan panjang n=[0,1,2,3,4,5,6,7], maka penentuan nilai *suffix*nya adalah :

P = 0
PA = 0,0
PAN = 0,0,0
PANP = 0,0,0,11,
karena X[3] = X[0]
PANPA = 0,0,0,1,2,2,
karena kombinasi X[3,4] = X[0,1]
PANPAN = 0,0,0,1,2,33,
karena X[3,4,5] = X[0,1,2]
PANPANC = 0,0,0,1,2,3,0
PANPANCA = 0,0,0,1,2,3,0,11,
karena X[7] = X[0]

Maka diperoleh nilai *suffix* untuk *pattern* X[n]= PANPANCA adalah 0,0,0,1,2,3,0,1.
Dari penjelasan di atas maka untuk *pattern* 'KIMIA', diperoleh:

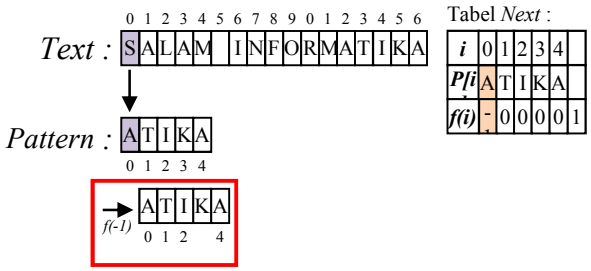
i	0	1	2	3	4
P[i]	A	T	I	K	A
f(i)	0	0	0	0	1

.... ***i*** merupakan panjang *pattern*.
.... ***P[i]*** merupakan karakter *pattern*.
.... ***f(i)*** merupakan nilai *suffix pattern*.

Dari sini, nilai untuk menggeser *pattern* telah diperoleh, namun agar proses penggeseran menjadi lebih mudah, perlu ditambah nilai -1 pada *suffix* f(i) ketika i=0. Ini berfungsi, ketika ketidakcocokan terjadi pada karakter pertama (i=0), maka *pattern* akan bergeser sebanyak 1 langkah ke kanan. Setelah ditambah nilai -1 pada f(0), maka kondisi tabel *next* saat ini yaitu:

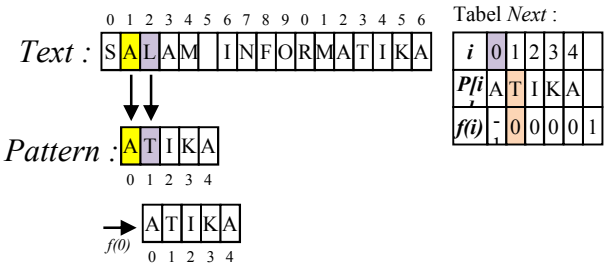
i	0	1	2	3	4	
P[i]	A	T	I	K	A	
f(i)	-1		0	0	0	1

Setelah mengetahui fungsi batasan yang dinyatakan dalam tabel *next*, algoritma *knuth morris pratt* kemudian mulai melakukan pencocokkan *string* dengan menempatkan *pattern* di awal *text*, sehingga i=j=0. Kemudian dari kiri ke kanan, algoritma *knuth morris pratt* mulai melakukan pencocokkan karakter per karakter pada *pattern* dengan karakter yang ada pada *text*.

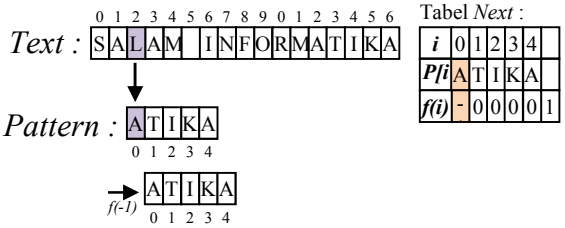


Karakter ‘A’ pada *pattern* tidak cocok dengan karakter ‘S’ yang ada pada *text*. Saat ketidak cocokan ditemukan, algoritma *knuth morris pratt* menggunakan fungsi batasan (*border function*) yang ditampilkan pada tabel *next* untuk menghitung sejauh mana pergeseran dilakukan.

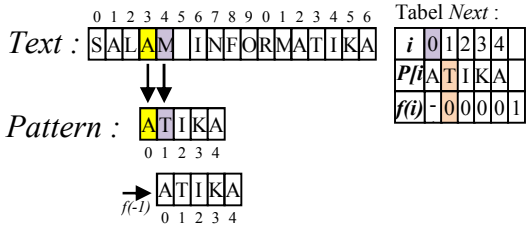
Pada tabel *next*, ketidakcocokan pada *pattern* terjadi pada karakter ‘A’ dengan nilai *suffix* f(0)=-1. Untuk nilai *suffix* f(0)=-1 maka *pattern* digeser satu langkah ke kanan. Maka posisi *pattern* saat ini yaitu :



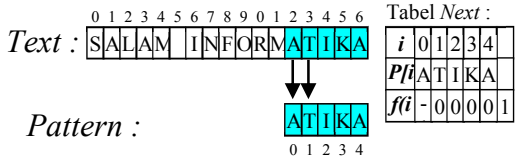
Algoritma *knuth morris pratt* kemudian mulai melakukan pencocokkan kembali. Dari posisi yang baru, dimana i=0, j=1, terjadi kecocokan pada karakter pertama, yaitu karakter ‘A’ pada *pattern* dengan karakter ‘A’ pada *text*. Lalu algoritma *knuth morris pratt* melakukan pencocokkan ke karakter selanjutnya, diketahui bahwa ketidakcocokkan terjadi saat j=2 pada karakter ‘L’ dengan i=1 pada karakter ‘T’. Lihat kembali ke tabel *next*. Ketidakcocokan terjadi pada karakter ‘T’ yang memiliki nilai *suffix* 0. *Knuth morris pratt* lalu mensejajarkan karakter yang tidak cocok tersebut dengan karakter yang ada di *pattern* saat i=0 yaitu karakter ‘A’, sehingga posisi *pattern* saat ini yaitu :



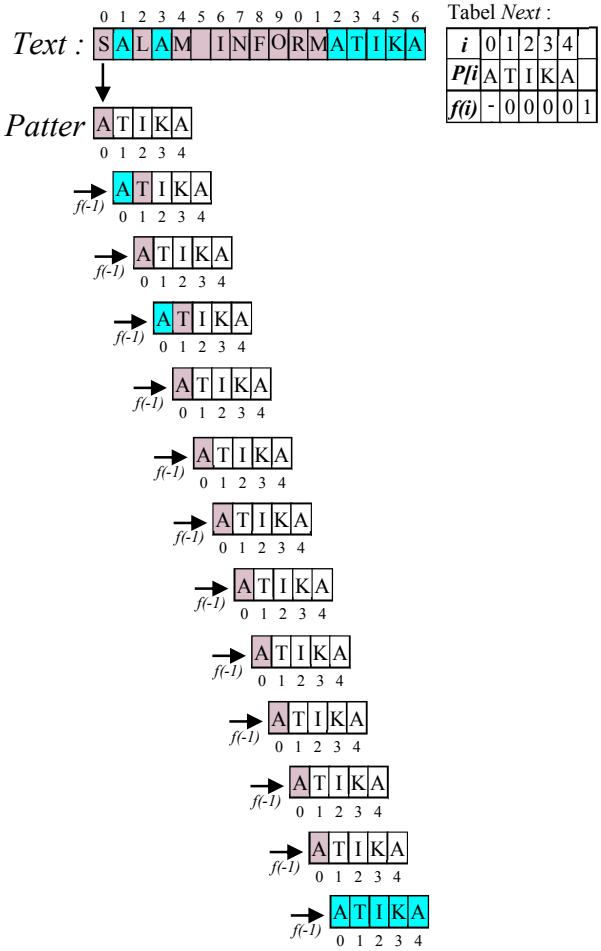
Selanjutnya, algoritma *knuth morris pratt* melakukan pencocokan kembali. Pada i=0 dan j=2 kembali terjadi kecocokan karakter, *knuth morris pratt* kemudian melakukan pencocokkan pada karakter selanjutnya. Diketahui ketidakcocokkan ditemukan pada i= lihat lagi nilai *suffix* pada tabel *next*. Ketidakcocokan terjadi pada karakter pertama, yaitu ‘A’ dengan nilai *suffix* -1, yang berarti *pattern* kembali digeser sebanyak 1 langkah ke kanan. Sehingga posisi *pattern* saat ini :



Dengan langkah yang sama seperti langkah di atas, algoritma *knuth morris pratt* akan terus melakukan pencocokan *string* hingga akhir *text*, atau hingga *pattern* yang dicari ditemukan. Hingga akhirnya pada i=0, dan j=2, karakter 'A' cocok. Kemudian algoritma *knuth morris pratt* melakukan pencocokan karakter per karakter pada *text* dan *pattern*, dan ternyata untuk *pattern*[i]=[0,1,2,3,4] cocok dengan karakter pada *text*[j]=[2,3,4,5,6], algoritma *knuth morris pratt* kemudian akan memberitahukan kecocokan pada posisi ini.



Adapun gambaran keseluruhan dari langkah pencocokan *string* di atas adalah sebagai berikut:



Metode *knuth morris pratt* dari awal proses hingga kecocokkan ditemukan. Dapat dilihat bahwa pergeseran karakter yang dilakukan metode *knuth morris pratt* tidak terlalu jauh, karena pada tabel *next*, terlihat bahwa *pattern* memiliki karakter yang beragam sehingga pergeseran karakter yang dilakukan tidak terlalu jauh. Sebaliknya, jika karakter pada *pattern* tidak terlalu beragam, maka akan terlihat sejauh apa metode *knuth morris pratt* mampu bergeser.

D. Implementasi Sistem

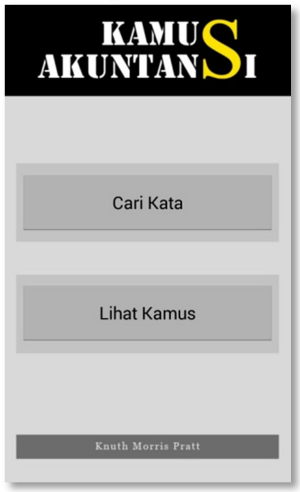
Tampilan awal untuk kamus istilah akuntansi menggunakan metode Knuth Morris Pratt

Gambar 2. Tampilan Splash Screen

Merupakan tampilan awal ketika pertama kali aplikasi dijalankan, *splash screen* berfungsi sebagai pemuat (*loader*) *database* yang digunakan.

1. Tampilan Menu Utama

Adapun tampilan menu utama kamus akuntasi adalah sebagai berikut:



Gambar 3. Tampilan Menu Utama

Merupakan tampilan ketika tampilan *splash screen* berakhir. Menu utama merupakan tampilan dimana semua fungsi utama berada.

2. Tampilan Proses Pencarian

Adapun tampilan proses pencarian menggunakan metode knuth morris pratt adalah sebagai berikut:

Gambar 4. (a) Tampilan awal sebelum proses pencarian dilakukan, dan (b) Tampilan hasil pencarian.

Merupakan tampilan ketika tombol Cari Kata yang ada pada menu utama ditekan. Tampilan pada Gambar 4a menunjukkan tampilan awal, ketika proses pencarian belum dimulai. Pada Gambar 4b menunjukkan hasil yang keluar saat kata yang dicari telah dimasukkan, dan kemudian tombol Cari ditekan. Pada tampilan inilah, metode pencarian dengan menggunakan metode *knuth morris pratt* dilakukan.

3. Tampilan Lihat Kamus

Isian dalam melihat isi data kamus adalah sebagai berikut :

(a) (b)

Gambar 5. (a) Tampilan awal saat tombol 'Lihat Kamus' ditekan, (b) Tampilan ketika istilah yang ada pada gambar (a) dipilih.

Tampilan yang muncul ketika tombol Lihat Kamus yang ada pada menu utama ditekan. Gambar 5a merupakan tampilan yang pertama muncul ketika tombol Lihat Kamus ditekan, Gambar 5a menampilkan daftar istilah-istilah dari *database*. Gambar 5b merupakan tampilan ketika salah satu istilah-istilah pada Gambar 5a dipilih.

IV. KESIMPULAN

Berdasarkan penelitian ini dapat disimpulkan bahwa:

1. Metode *knuth morris pratt* menggunakan fungsi batasan (*border function*) untuk menentukan jumlah pergeseran yang dilakukan.
2. Pengaruh *suffix* (kesamaan karakter yang ada pada *pattern*) mempengaruhi kompleksitas waktu yang dibutuhkan metode *knuth morris pratt*, karena semakin beragam karakter pada *pattern* maka waktu yang dibutuhkan untuk mencari kecocokan tergolong lambat.
3. Aplikasi kamus istilah akuntansi sangat membantu dalam melakukan pencarian, dengan adanya aplikasi kamus istilah, proses pencarian dapat dilakukan dengan lebih mudah dan cepat.

V. SARAN

1. Penggunaan metode *knuth morris pratt* menjadi tidak sempurna jika penggunaan karakter pada *pattern* beragam. Untuk aplikasi kamus istilah akuntansi yang memiliki keberagaman karakter, metode pencocokan *string* lainnya seperti *boyer moore* lebih cocok digunakan.
2. Metode *knuth morris pratt* lebih cocok digunakan pada pencarian yang tidak banyak menggunakan karakter yang beragam.

REFERENSI

[1] Kurniasih. dan Ema Utami. 2014. "Aplikasi Kamus Istilah Komputer Berbasis Android". STMIK Amikom Yogyakarta.

[2] Reza Zulman, Muhammad. 2013. *Aplikasi Kamus Istilah Pemograman Bahasa C Menggunakan Metode Binary Search dan Approximate String Matching Pada Platform Android*. Tugas Akhir. Program Studi Teknik Informatika, Politeknik Negeri Lhokseumawe.

[3] Wahana Komputer, Tim. 2013. *Step by Step Menjadi Programmer Android*, Yogyakarta : Andi dan Semarang : Wahana Komputer.

[4] Safaat H, Nazruddin. 2014. *Pemograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android*. Edisi Revisi ke Dua. Bandung : INFORMATIKA.

[5] Satyaputra, Alfa. dan Eva Maulina Aritonang. 2012. *Java for Benginners with Eclipse 4.2 Juno*, Jakarta : Elex Media Komputindo.

[6] Rossa A. dan M. Shalahuddin. 2013. *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*, Bandung : INFORMATIKA.

[7] Wachid Kusuma, Muhammad. 2013. "Pencocokan String Dalam Fitur Autocompletion Pada Text Editor atau Integrated Development Environment (IDE)". Institut Teknologi Bandung.

[8] Hartoyo, Eko, Yus Vembrina, dkk. 2010. "Analisis Algoritma Pencarian String (String Matching)". URL: http://www.docstoc.com/docs/40337710/Analisis-Algoritma-Pencarian- String-(-String-Matching-). Diakses tanggal 12 februari 2015.

[9] Rama Aulia . 2008. Analisa Algoritma Morris Pratt dan Algoritma Boyer Moore dalam Proses Pencarian String